題目所付的 Inputs 為 sample Inputs，機考的 input 會有不同，請就題目的描述來實作

CCU CSIE
Data Structure Course - 2020 Fall
Computer-Based Test I Questions

Instructor: Yu-Ling Hsueh

TA: Ben, Sine, Edward, and Peter

Note: You should pay attention to the newline characters in the outputs.

1. Implement sparse matrix transpose and multiplication

Given two sparse matrices A and B below, please implement sparse matrix multiplication to compute A*B using the **fast transpose** approach introduced in the lecture. The input consists of the first matrix followed by the second matrix and the first line for each matrix contains the number of rows and columns of the matrix. You must transpose matrix B before multiplying it by A. First, output the result of the starting position of each column in a sequence of numbers when performing the fast transpose. Subsequently, output the multiplication result for non-zero terms only with the first line to represent the number of rows, columns, and non-zero terms.

Note: You must define a triple struct <row, col, value> and use an 1-D array to store non-zero terms in the matrix.

$$\underset{A}{\begin{bmatrix} 0 & 3 & 0 & 1 \\ 0 & 1 & 3 & 4 \\ 0 & 0 & -2 & 0 \end{bmatrix}} \times \underset{B}{\begin{bmatrix} 3 & 1 \\ -1 & 0 \\ 5 & 0 \\ 0 & 2 \end{bmatrix}} = \begin{bmatrix} -3 & 2 \\ 14 & 8 \\ -10 & 0 \end{bmatrix}$$

Test Case

Please test your program with Input, and then check the answers with Output.

Listing 1: Implement sparse matrix multiplication.

```
Input：
3 4
0 3 0 1
0 1 3 4
0 0 -2 0
4 2
3 1
-1 0
5 0
0 2
Output：
Starting position for each column of B:
1 4
A * B:
3 2 5
0 0 -3
0 1 2
1 0 14
1 1 8
2 0 -10
```

2. KMP

Please read the given file that contains the NAME and INDEX data fields. At first, you should initialize MAIN_STRING to 'NULL'. You then use MAIN_STRING to store the data reading from the file. The value of NAME is the data you should insert into MAIN_STRING, and INDEX represents the index position where you should insert a new string at the following insertion. For instance, after the initialization, you read 'Hank' and '1' from the file; hence, MAIN_STRING is 'Hank'. Then, you keep reading 'Henry' and '2' from the file, and you should insert 'Henry' to the '1st' position of the previously inserted data (i.e., Hank) in MAIN_STRING. Therefore, MAIN_STRING contains 'HHenryank'. The line "P:ffaaf" with a prefix "P" in the input file is a pattern to be searched. After performing all the string concatenations, the final text is stored in MAIN_STRING. Next, implement the KMP algorithm to search for the pattern in MAIN_STRING. You must print the final string (i.e., MAIN_STRING), the entire content of LPS [], the sum of LPS [] which is used to skip characters while matching, and the index position of the first occurrence of the matched word found in MAIN_STRING.

Test Case

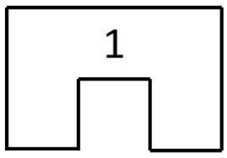
Please test your program with Input, and then check the answers with Output.

Listing 2: KMP

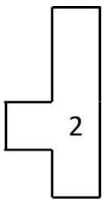
```
Input：
Hank,1
Henry,2
WenJ,3
Cody,4
ffaaf,4
P:ffaaf
Output：
Final string: HHWCffaafodyenJenryank
Lps = {0,1,0,0,1}
Sum = 2
Index = 4
```

3. Puzzle 2

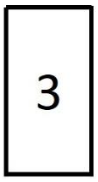
The problem is to judge whether the given puzzles can be fully merged on a four by four grid. The number of puzzles could be one up to four. All puzzles cannot be rotated or flipped, and must be used to form a four by four grid. Each puzzle is numbered by the order it appears in the input. That is, the first puzzle is 1, the next is 2, etc. Figures (a) – (d) show the examples of the given four puzzles. All puzzles are no larger than four by four grid. The result could be one solution or no solution. Figure (e) shows a successful merge.



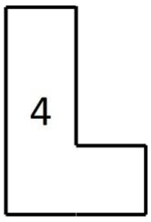
(a) Puzzle 1



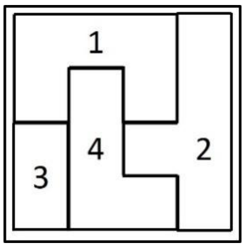
(b) Puzzle 2



(c) Puzzle 3



(d) Puzzle 4



(e) A successful merge

Puzzles and the solution

The first line of the input is the total number of puzzles. Subsequently, each puzzle input contains the number of rows and the number of columns, followed by the shape of the puzzle. If they can be merged, you need to output the merged result. For puzzles which have no possible solution, simply report 'NO' for the answer.

Test Case

Please test your program with Input, and then check the answers with Output.

Listing 3: Puzzle 2

Input：

```
4
2 3
111
101
4 2
02
02
22
02
2 1
3
3
3 2
40
40
```

```
44
Output：
1112
1412
3422
3442
```

4. Infix to postfix conversion

Please finish a program which can change the expression from the infix form to the postfix form in C using a stack. The input operands are lowercase letters「*a*」to「*z*」; meanwhile, the input operators are 「(」,「)」,「*」,「/」,「+」, and 「-」. Furthermore, we have two types of the precedence of operators. An in-stack precedence is 「)」>「*」=「/」>「+」 =「-」>「(」 and an incoming precedence is 「(」>「)」>「*」=「/」>「+」 =「-」. The first line of the input is the number of expressions, and the second line to the last line in the input are the expressions. The output needs to display the result of the postfix expression.

Test Case

Please test your program with Input and Output.

Listing 4: Infix to postfix conversion.

```
Input：
2
a+b*c
a*(b+c)*d
Output：
abc*+
abc+*d*
```

5. Postfix expression evaluation

Please finish a program which can evaluate postfix expression in C using a stack. The input is a postfix expression. The input operands are unit digits「*1*」to「*9*」; the input operators are「*」,「/」,「+」, and「-」. As postfix expression is without parenthesis and can be evaluated as two operands and an operator at a time, this becomes easier for the complier and the computer to handle. The first line of the input is the number of expressions, and the second line to the last line in the input are the expressions. Each output for one expression includes two results: (1) the result of evaluating postfix expression (2) the maximum number of elements in the stack during the computation.

Test Case

Please test your program with Input, and then check the answers with Output.

Listing 5: Postfix expression evaluation.

```
Input：
2
62/3–42*+
33/4-16*+52*-
Output：
8
3
-7
3
```

6. Text Decoder

Given an encoded string, please decode it into the original string. The rule is that k[s] represents s exactly repeating k times, where k must be a positive integer, and *s* consists of one or more alphabets. In the input string, there is no white space, and all square brackets are well-formed.

Test Case

Please test your program with Input, and then check the answers with Output. The first line represents the number of encoded strings.

Listing 6: Text decoder

```
Input：
4
3[a]2[bc]
3[a2[c]]
2[abc]3[cd]ef
abc3[cd]xyz
```

Output：

```
aaabcbc
accaccacc
abcabccdcdcdef
abccdcdcdxyz
```

7. Maze

Given a 10 by 10 maze with one entrance and one exit, find the way to the exit (8,8) from the entrance (1,1). There is a border (represented by 1s) around the maze. There are only four moving directions: right > down > left > up. You must use a stack to store the moving path. Input is a 10 by 10 maze with a border. Output must be the moving path that consists of steps separate by “,”.

Test Case

Please test your program with Input, and then check the answers with Output.

Listing 7: Maze

Input：

```
1111111111
1011000011
1000011111
1011111111
1000111111
1110111111
1110111111
1110111111
1110000001
1111111111
```

Output：

```
(1,1),(2,1),(2,2),(2,3),(2,4),(1,4),(1,5),(1,6),(1,7),(1,6),(1,5),(1,4),(2,4),(2,3),
(2,2),(2,1),(3,1),(4,1),(4,2),(4,3),(5,3),(6,3),(7,3),(8,3),(8,4),(8,5),(8,6),(8,7),(8,8)
```

8. Addition operation of the polynomials

In mathematics, a polynomial is an expression consisting of variables and coefficients, and involves the operations of addition, subtraction, multiplication, and non-negative integer exponentiation of variables. An example of a polynomial of a single indeterminate x is $x^2 - 4x + 7$. Given a sequence of integers that represent two polynomials, you are to write a program to perform the addition operation and output the addition result of two polynomials.

The pair of integers in the first row of the input represents the coefficient and the exponent of a polynomial term. The symbol "next" is used to separate the first polynomial and the second polynomial followed by a tag "end." For example, the first polynomial is shown as below:

$3x^3+4x^2+5x^1$

And following the line with the symbol "next", there is the second polynomial which is shown as below:

$3x^3+5$

When reading the tag "end" in the input, the program starts to calculate the answer. Finally, the result of the addition operation is

$6x^3+4x^2+5x^1+5$

In your output, you must convert it to the output format as follows:

```
6 3
4 2
5 1
5 0
```

Note: You must use linked lists to implement the operation; otherwise, no points will be given. All the polynomials are ordering in descending order.

Test Case

Please test your program with Input, and then check the answers with Output.

Listing 8: Addition operation of the polynomials

Input：

```
3 3
4 2
5 1
next
3 3
5 0
end
```

Output：

```
6 3
4 2
5 1
5 0
```

9. Invert a singly linked list

Please implement a function to invert a singly linked list in C. The input includes two lines: a sequence of input numbers, and a sequence of partition codes. The number of the input numbers is multiplication by 9 (e.g., 9, 18, 27,…), so that we can divide the numbers into three partitions. We define 1 as the front partition, 2 as the middle partition, and 3 as the rear partition. As the example shown in the following input, there are 9 numbers so each partition has three numbers (3 4 8) (9 11 22) (13 99 23). You must read each partition code sequentially and invert the partition. After that, insert the inverted partition into the back of the sequence of numbers. For example, the first partition code is 1, the front partition (3 4 8) is inverted to (8 4 3) and insert them into the back of the numbers. So the sequence of numbers becomes 9 11 22 13 99 23 8 4 3. Read the next partition code and continue the operation based on the output of the previous the partition-inversion-insertion operation. You must output each list of numbers for each operation.

Note: You must use a linked list; otherwise, no points will be given.

Test Case

Please test your program with Input, and then check the answers with Output.

Listing 9: Invert a singly linked list

Input：

```
3 4 8 9 11 22 13 99 23
1 2 3 3 1 2
```

Output：

```
9 11 22 13 99 23 8 4 3
9 11 22 8 4 3 23 99 13
9 11 22 8 4 3 13 99 23
9 11 22 8 4 3 23 99 13
8 4 3 23 99 13 22 11 9
```

```
8 4 3 22 11 9 13 99 23
```

10. Implement a stack by using linked lists

Please finish a program which can push and pop a stack by using a linked list in C. The input includes a sequence of push and pop operations. The output includes two lines of the results: (1) the size of the stack, and (2) the status of the stack from the bottom to the top.

Note: You must use linked list; otherwise, no points will be given.

Test Case

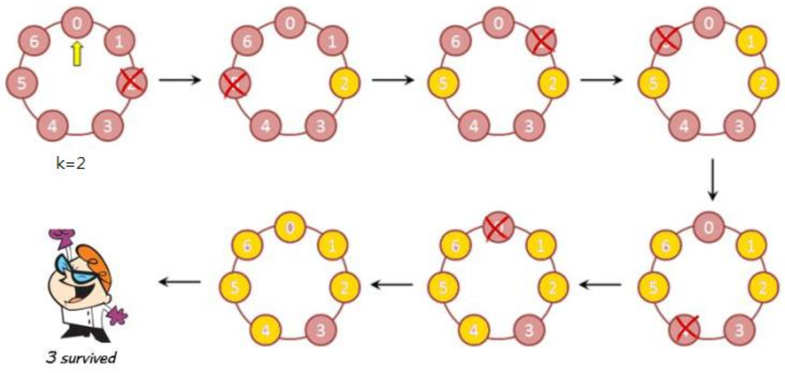
Please test your program with Input, and then check the answers with Output.

Listing 10: Implement a stack by using a linked list

```
Input：
Push apple
Push peach
Push mango
Pop
Push orange
Push papaya
Pop
Pop
Push lemon
Output：
3
apple, peach, lemon
```

11. Circular linked list

Please implement a program which creates a **circular linked list** and displays the elements in the list. Next, use it to solve the Josephus problem. The illustration of the processing about Josephus problem is the following figure. Assume there are eight people. Each number represents a person. *k* is used as a specified number of people to be skipped. Therefore, in the first round, #2 is killed. The procedure is repeated with the remaining people, starting with the next person who is not dead, going in the **same direction** and skipping the same number of people, until only one person remains, and is survived. The default direction is **clockwise**. The input file contains a list of names separated by commas. The second line is the *k* value. The procedure begins with the first name in the input list. Please output the sequence of the people who are killed during the process.



An illustration of the Josephus problem.

Note: You must use linked lists; otherwise, no points will be given.

Test Case

Please test your program with Input, and then check the answers with Output.

Listing 11: Circular linked list

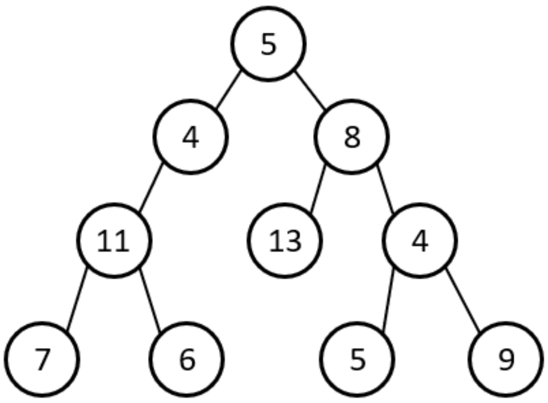
```
Input：
Jay, Tom, Wayne, Kevin, Helen, Jim, Kile, Ruby, Nick, Patty, Mike, Kurt
1

Output：
Tom is killed.
Kevin is killed.
Jim is killed.
Ruby is killed.
Patty is killed.
Kurt is killed.
Wayne is killed.
Kile is killed.
Mike is killed.
Helen is killed.
Jay is killed.
```

```
Nick is survived.
```

12. Path sum

In addition to implementing a binary tree, you need to implement a path sum function to find all root-to-leaf paths whose sum is equal to the given sum.



Example of a binary tree

The first line of the input represents a tree diagram as illustrated in the above figure. The symbol "x" represents a null. The second line of the input data is the given sum. If you cannot find a path that its sum equals the given sum, you need to print "Not Found" in the output.

Test Case:

Please test your program with Input, and then check the answers with Output.

Listing 12: Path sum

Input 1 :

```
5 4 8 11 x 13 4 7 6 x x x x 5 9\n
26
```

Output 1 :

```
5 4 11 6\n
5 8 13\n
5 8 4 9
```

Input 2:

```
5 4 8 11 x 13 4 7 6 x x x x 5 9\n
28
```
Output 2:
```
Not Found
```

13. Implement a binary tree traversal with linked lists

Please implement a binary search tree with linked lists and traversal with in-order, post-order, pre-order. The input consists of a sequence of numbers. You must build a binary search tree according to the input order. The output includes three lines of the results: (1) the in-order traversal, (2) the post-order traversal, and (3) the pre-order traversal.

Note: You must use linked lists; otherwise, no points will be given.

Test Case

Please test your program with Input, and then check the answers with Output.

Listing 13: Implement a binary tree traversal with linked lists

Input：
```
4 6 8 9 12 5 7
```
Output：
```
4 5 6 7 8 9 12
5 7 12 9 8 6 4
4 6 5 8 7 9 12
```

14. Heap sort

Given a sequence of n numbers, build a heap according to the input order using an array and implement a heap sort that results in a set of sorted numbers in ascending order. The output includes: (1) the contents of the array that stores the heap in each step, and (2) the sorted numbers in ascending order.

Test Case

Please test your program with Input, and then check the answers with Output.

Listing 14: Heap sort

Input:
```
1 4 5 0 6 7 2
```
Output：
```
0 1 2 4 6 7 5
1 4 2 5 6 7
2 4 7 5 6
4 5 7 6
5 6 7
6 7
7
0 1 2 4 5 6 7
```

15. Operations of a binary search tree

Implement a binary search tree with insert(x) and delete(x) functions using an array. You must build a binary search tree according to the input order. The input consists of a sequence of numbers in the first line. The insert/delete function and the number to be inserted/deleted are shown in the remaining lines. For the delete operation, you must first consider the node with the largest key from the left sub-tree to replace the deleted node. Finally, print out the entire tree in level order from the top to the down. If there are more than two numbers at the same level, you should print out the numbers from the left to the right

Test Case

Please test your program with Input, and then check the answers with Output.

Listing 15: Operations of a binary search tree

Input:
```
4 5 1 0 6 7 2
insert(44)
delete(6)
insert(3)
delete(2)
```
Output:
```
4 1 5 0 3 7 44
```